Summer 6-1-1986

Solving Systems of Nonlinear Equations by Tensor Methods ; CU-CS-334-86

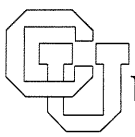
Robert B. Schnabel
University of Colorado Boulder

Paul D. Frank
University of Colorado Boulder

Solving Systems of Nonlinear Equations By Tensor Methods

Robert B. Schnabel and Paul D. Frank*

CU-CS-334-86 June 1986



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

* Research supported by NSF grant DCR-8403483 ARO contract DAAG 29-84-K-0140

Abstract

Tensor methods are a class of general purpose methods for solving systems of nonlinear equations. They are especially intended to efficiently solve problems where the Jacobian matrix at the solution is singular or ill-conditioned, while remaining at least as efficient as standard methods on nonsingular problems. Their distinguishing feature is that they base each iteration on a quadratic model of the nonlinear function. The model has a simple second order term that allows it to interpolate more information about the nonlinear function than standard, linear model based methods, without significantly increasing the cost of forming, storing, or solving the model.

This paper summarizes two types of tensor methods, derivative tensor methods that calculate an analytic or finite difference Jacobian at each iteration, and secant tensor methods that avoid Jacobian evaluations. Both are shown to require no more function or derivative information per iteration, and hardly more storage or arithmetic operations per iteration, than standard linear model based methods. Computational results are presented that indicate that both tensor methods are consistently at least as reliable as the corresponding linear model based methods, and are significantly more efficient, both on nonsingular and on singular test problems.

1. Introduction

This paper summarizes a recently developed class of methods, called tensor methods, for solving the nonlinear equations problem

$$\text{given } F : R^n \to R^n, \quad \text{find } x_* \in R^n \text{ such that } F(x_*) = 0 \tag{1.1}$$

where it is assumed that $F(x)$ is at least once continuously differentiable. Tensor methods are especially intended to efficiently solve problems where the Jacobian matrix of F at x_*, $F'(x_*) \in R^{n \times n}$, is singular or ill-conditioned. They also are intended to be at least as efficient as standard methods on problems where $F'(x_*)$ is nonsingular. Their distinguishing feature is that they base each iteration on a quadratic model of $F(x)$ whose second order term has a simple form.

Systems of nonlinear equations arise frequently in many practical applications including equilibrium calculations, curve tracing problems, and as subproblems in solving nonlinear systems of differential equations. In many important situations, $F'(x_*)$ is singular or ill-conditioned. For example, in some stiff systems of ordinary differential equations the Jacobian of the associated system of nonlinear equations is nearly singular for all x. The calculation of turning points in curve tracing problems and the solution of over-parameterized data fitting problems are other common situations that lead to singular systems of equations. In all these cases, it is important to notice that the (near) rank deficiency in the derivative matrix usually is small. This is the case in which our methods are intended to improve upon standard methods.

Standard methods for solving (1.1) base each iteration upon a linear model $M(x)$ of $F(x)$ around the current iterate $x_c \in R^n$,

$$M(x_c+d) = F(x_c) + J_c d \tag{1.2}$$

where $d \in R^n$, $J_c \in R^{n\times n}$. These methods can be divided into two classes: derivative methods, where J_c is the current Jacobian matrix $F'(x_c)$ or a finite difference approximation to it, and secant methods, where J_c is a secant (quasi-Newton) approximation to the Jacobian. For a general description of these methods, see e.g. Dennis and Schnabel [1983].

When the analytic Jacobian is available, the linear model (1.2) becomes

$$M(x_c+d) = F(x_c) + F'(x_c)d. \tag{1.3}$$

The standard method for nonlinear equations, Newton's method, consists of setting the next iterate x_+ to the root of (1.3),

$$x_+ = x_c - F'(x_c)^{-1}F(x_c). \tag{1.4}$$

If $F'(x_c)$ is Lipschitz continuous in a neighborhood containing the root x_* and $F'(x_*)$ is nonsingular, then the sequence of iterates produced by (1.4) converges locally and q-quadratically to x_*. This means that there exist $\delta>0$ and $c\geq 0$ such that the sequence of iterates $\{x_k\}$ produced by Newton's method obeys

$$\|x_{k+1} - x_*\| \leq c\ \|x_k - x_*\|^2$$

if $\|x_0 - x_*\| \leq \delta$. In practice, local q-quadratic convergence means eventual fast convergence.

Newton's method usually is not quickly locally convergent, however, if $F'(x_*)$ is singular. For example when applied to one equation in one unknown ($n=1$) where $f'(x_*)=0$ but $f''(x_*)\neq 0$, Newton's method is locally q-linearly convergent with constant converging to ½, meaning that the sequence of iterates $\{x_k\}$ obeys

$$|x_{k+1} - x_*| = c_k\ |x_k - x_*|\ , \quad \lim_{k\to\infty} c_k = \tfrac{1}{2}$$

if $|x_0 - x_*|$ is sufficiently small. For systems of equations, the situation is more complex and

has been analyzed by many authors, including Decker and Kelley [1980a, 1980b, 1982], Decker, Keller, and Kelley [1983], Griewank [1980a, 1980b, 1985], Griewank and Osborne [1981,1983], Keller [1970], Kelley and Suresh [1983], Rall [1966], and Reddien [1978, 1980]. In summary, their papers show that from many starting points, Newton's method for systems of equations also is locally q-linearly convergent with constant converging to ½, although for some problems with starting points arbitrarily close to x_*, (1.4) may be undefined or lead further away from the solution (see e.g. Griewank and Osborne [1983]). In practice, Newton's method usually exhibits local linear convergence with constant $\cong$ ½ on singular problems, much slower convergence than one would like.

When analytic derivatives are unavailable and function evaluation is expensive, (1.1) generally is solved by a secant method. These methods attempt, as much as possible, to solve (1.1) using only the function values at the iterates. The model (1.2) still is used but the matrix J_c is generated from these function values and may be a very rough approximation to $F'(x_c)$. In the most commonly used secant method for systems of equations, Broyden's method, the Jacobian approximation J_c is chosen to be the smallest change to the previous Jacobian approximation which causes the new linear model $M(x)$ to interpolate the value of $F(x)$ at the previous iterate. This results in a rank one change to the Jacobian approximation at each iteration. (The details are given in Section 3.1.) The initial Jacobian approximation is made by finite differences, and sometimes it is necessary to reset J_c to a finite difference approximation at subsequent iterations.

The sequence of iterates produced by Broyden's method converges locally and q-superlinearly to x_* as long as $F'(x_c)$ is Lipschitz continuous in a neighborhood containing the root x_* and $F'(x_*)$ is nonsingular (Broyden, Dennis, and Moré [1973]). This means that there exist $\delta>0$ and $r>0$ such that the sequence of iterates $\{x_k\}$ obeys

$$\lim_{k \to \infty} \quad \|x_{k+1} - x_*\| \; / \; \|x_k - x_*\| = 0$$

if $\|x_0 - x_*\| \leq \delta$ and $\|J_0 - F'(x_0)\| \leq \tau$. In practice, secant methods still are quickly convergent on nonsingular problems, and while they usually require more iterations than Newton's method, they usually require fewer function evaluations than a finite difference implementation of Newton's method.

However secant methods, like Newton's method, are slowly convergent on problems where $F'(x_*)$ is singular. For example on one variable problems with $f'(x_*)=0$ but $f''(x_*)\neq 0$, the secant method is locally q-linearly convergent with constant converging to 0.618, a slightly slower rate than Newton's method. For multiple variable problems with $\text{rank}(F'(x_*)) = n-1$, Decker and Kelley [1985] have shown that this same rate of convergence is obtained by Broyden's method from certain starting points. As in the case of Newton's method, this slow linear convergence usually is observed in practice, making quicker methods desirable.

Several papers, for example Decker and Kelley [1982], Decker, Keller, and Kelley [1983], Griewank [1980a, 1985], Kelley [1985], and Kelley and Suresh [1983], propose methods that are rapidly convergent on some singular problems. Many of these methods are related to the one dimensional acceleration technique of taking j times the Newton step if one has a root of multiplicity j. Some other methods explicitly calculate and use higher derivative information in null space directions. To our knowledge, no computational experience with a complete method of this type has been published, and it is not clear how amenable these techniques are to solving general systems of nonlinear equations where it is unknown a priori whether $F'(x_*)$ is singular or not.

The major aim of tensor methods is to provide general purpose methods that have rapid convergence even when $F'(x_*)$ is singular. In addition, the methods should not experience any special difficulty when J_c is singular or ill-conditioned, while methods based on (1.2) must be

modified in this case.

Tensor methods are based on expanding the linear model (1.2) of $F(x)$ around x_c to the quadratic model

$$M_T(x_c+d) = F(x_c) + J_c d + \tfrac{1}{2} T_c dd \tag{1.5}$$

where $T_c \in R^{n\times n\times n}$ and J_c is $F'(x_c)$ or a secant approximation to it. The three dimensional object T_c often is referred to as a tensor, hence we call (1.5) a *tensor model*, and methods based upon (1.5) *tensor methods*. The term $T_c dd$ is defined by $(T_c dd)[i] = d^T H_i d$, where H_i is the i^{th} horizontal face of T_c. Thus the model $M_T(x_c+d)$ is the n-vector of quadratic models of the component functions of $F(x)$,

$$(M_T(x_c+d))[i] = f_i + g_i^T d + \tfrac{1}{2} d^T H_i d \quad , \quad i = 1, \cdots, n$$

where $f_i = F(x_c)[i]$, g_i^T = row i of $F'(x_c)$ or an approximation to it, and H_i is (an approximation to) the Hessian matrix of the i^{th} component function of $F(x)$.

The obvious choice of T_c in (1.5) is the matrix $F''(x_c)$ of second partial derivatives of F at x_c; if J_c is $F'(x_c)$, this makes (1.5) the first three terms of the Taylor series expansion of F around x_c. Several serious disadvantages, however, make (1.5) with $T_c = F''(x_c)$ unacceptable for algorithmic use. First, the n^3 second partial derivatives of F at x_c would have to be computed at each iteration. Second, the model would take more than $n^3/2$ locations to store as compared to the n^2 locations for the standard model. Third, to find a root of the model, at each iteration one would have to solve a system of n quadratic equations in n unknowns, which for $n > 1$ requires an iterative procedure. Finally, the model might not have a real root.

To use a model of form (1.5) and avoid these disadvantages, our tensor methods use a very restricted form of T_c. In particular, our tensor methods require no additional derivative or function information; the additional costs of forming and solving the tensor model are small compared to the $O(n^3)$ arithmetic cost per iteration of standard methods; and the additional

storage required for our tensor models is small compared to the n^2 storage required for the Jacobian. The remainder of this paper describes how we utilize the tensor term T_c in the model (1.5) and what benefits we obtain from its inclusion. In Section 2 we summarize the use of a tensor model in derivative methods for nonlinear equations, and our computational experience with this method. Section 3 similarly presents the use of and computational experience with a tensor model in secant methods for nonlinear equations. In Section 4 we briefly comment on extensions of tensor methods to nonlinear least squares and to unconstrained optimization. More details on this research can be found in Frank [1984], Schnabel and Frank [1984], and an upcoming paper by Frank and Schnabel.

Notice that we are denoting members of a sequence of n-vectors x by $\{x_k\}$ where each $x_k \in R^n$, and components of a vector $v \in R^n$ by $v[i] \in R$.

2. Derivative tensor methods

Derivative tensor methods base an algorithm for solving systems of nonlinear equations on a model of the form

$$M_T(x_c+d) = F(x_c) + F'(x_c)d + \tfrac{1}{2}T_c dd, \tag{2.1}$$

where it is assumed that $F'(x_c)$ either is supplied analytically or is calculated by finite differences. Their aim is to choose $T_c \in R^{n \times n \times n}$ so that the model (2.1) is hardly more expensive to form, store, or solve than the standard model (1.3), while still leading to an algorithm that requires fewer function evaluations than standard methods to solve difficult problems.

2.1 Forming the tensor model

The first step in deriving a method based on (2.1) is to choose the second order term T_c. We do not use any second derivative information in constructing T_c. Instead, we construct the second order term in (2.1) by asking the model to interpolate additional values of the function $F(x)$ that have already been computed by the algorithm. In particular, we ask the model to satisfy

$$F(x_{-k}) = F(x_c) + F'(x_c)s_k + \tfrac{1}{2}T_c s_k s_k, \quad k=1, \cdots, p \tag{2.2a}$$

where

$$s_k = x_{-k} - x_c, \quad k=1, \cdots, p \quad . \tag{2.2b}$$

and $x_{-1}, \cdots, x_{-p}$ are some set of p not necessarily consecutive past iterates.

For the equations (2.2) to be consistent, the past points $\{x_{-k}\}$ must be selected so that set of directions $\{s_k\}$ is linearly independent. In fact, we enforce a far more restrictive condition. We always set x_{-1} to the most recent iterate. We then include each remaining past iterate in the set of points to be interpolated if the step from it to x_c makes an angle of at least Θ degrees with the subspace spanned by the steps to the already selected more recent iterates. Here Θ is some fixed angle between 20 and 45 degrees. In addition, we consider at most $\sqrt{n}$ past iterates. The bounds $\sqrt{n}$ and 20-45 degrees have been shown by computational experience to be reasonable. This procedure for selecting past iterates to interpolate is implemented easily using a modified Gram-Schmidt algorithm, and requires about n^2 multiplications and additions.

The equations (2.2) are a set of $np \leq n^{1.5}$ linear equations in the n^3 unknowns comprising T_c. Thus T_c is underdetermined, so we follow the standard and successful practice in secant methods for nonlinear equations and optimization (see e.g. Dennis and Schnabel [1979]) and choose T_c to be the solution to

$$\underset{T_c \in R^{n\times n\times n}}{\text{minimize}} \quad \| T_c \|_F \tag{2.3}$$

$$\text{subject to} \quad T_c s_k s_k = t_k = 2\,(\,F(x_{-k}) - F(x_c) - F'(x_c)s_k\,)\,, \quad k=1, \cdots, p$$

where $\|\cdot\|_F$ is the Frobenius norm. If we denote by uvw the rank one tensor whose i^{th} horizontal face is the rank one matrix $u[i](vw^T)$, then the solution to (2.3) is shown by Schnabel and Frank [1984] to be

$$T_c = \sum_{k=1}^{n} a_k\, s_k\, s_k. \tag{2.4}$$

where $(a_1[i], \cdots, a_p[i])^T = M^{-1} * (t_1[i], \cdots, t_p[i])^T$, $i=1, \cdots, n$, with $M \in R^{p\times p}$ the positive definite matrix defined by $M[i,j] = (s_i^T s_j)^2$, $1 \le i,j \le p$.

Substituting (2.4) into the tensor model (2.1) gives

$$M_T(x_c + d) = F(x_c) + F'(x_c)d + \tfrac{1}{2} \sum_{k=1}^{p} a_k\, (d^T s_k)^2. \tag{2.5}$$

The simple form of the second order term in (2.5) is the key to being able to efficiently form, store, and solve the tensor model. Since $p \le \sqrt{n}$, the additional storage for the entire method is at most $4\sqrt{n}$ n-vectors, compared to the n^2 storage for $F'(x_c)$. The cost of forming the tensor model by the above procedure is at most $n^{2.5}$ multiplications and additions per iteration, small compared to the at least $n^3/3$ multiplications and additions per iteration required by standard methods.

2.2 Solving the derivative tensor model

To base an efficient algorithm on the tensor model (2.5), we need to efficiently find a root of this model, that is a $d \in R^n$ for which

$$M_T(x_c+d) = F(x_c) + F'(x_c)d + \tfrac{1}{2} \sum_{k=1}^{p} a_k \, (d^T s_k)^2 = 0. \tag{2.6}$$

In some cases, the tensor model may have no root; it is then appropriate to choose d to minimize the tensor model in some norm. We choose the l_2 norm, so that the general problem we wish to solve is to choose $d \in R^n$ to minimize $\|M_T(x_c+d)\|_2$.

The basic idea behind efficiently solving (2.6) is that since M_T only is quadratic on the p dimensional subspace spanned by $\{s_k\}$, and is linear on the orthogonal complement to this subspace, it may be possible to solve (2.6) by solving a system of p quadratic equations in p unknowns plus a system of $n-p$ linear equations in $n-p$ unknowns. This is accomplished by a procedure given in Schnabel and Frank [1984]. This procedure first makes an orthogonal transformation of the variable space to $\hat{d} = Q^T d$, so that all n equations are quadratic only in the last p components of $\hat{d}$, $\hat{d}_2 \in R^p$, and are linear in the first $n-p$ components, $\hat{d}_1 \in R^{n-p}$. It then makes an orthogonal transformation of the equations that eliminates the linear variables $\hat{d}_1$ from the final p (actually $q \geq p$, see below) equations and makes the preceding equations triangular in $\hat{d}_1$. The result is $n-q$ equations that are linear in the $n-p$ variables $\hat{d}_1$,

$$\tilde{F}_1 + \tilde{J}_1 \hat{d}_1 + \tilde{J}_2 \hat{d}_2 + \tfrac{1}{2} \tilde{A}_1 \{\hat{S}_2^T \hat{d}_2\}^2 = 0 \tag{2.7a}$$

where $\tilde{J}_1$ is upper triangular, plus the system of q quadratic equations in the p unknowns $\hat{d}_2$

$$\tilde{F}_2 + \tilde{J}_3 \hat{d}_2 + \tfrac{1}{2} \tilde{A}_2 \{\hat{S}_2^T \hat{d}_2\}^2 = 0. \tag{2.7b}$$

Here $q \geq p$, with $q=p$ as long as J is nonsingular, or J is singular but J augmented by the p rows $\{s_k^T\}$ has full column rank. In practice this means that q generally equals p unless $\text{rank}(F'(x_c)) < n-p$. The root or minimizer of M_T is then found from (2.7) by calculating the $\hat{d}_2$ which is the root of minimizer of the quadratic system of equations (2.7b), substituting this $\hat{d}_2$ into (2.7a) and calculating $\hat{d}_1$ by solving a triangular system of linear equations, and multiplying $\hat{d}$ by Q to obtain d.

The cost of solving the tensor model by this process is the standard $2/3\ n^3$ cost of a QR factorization, plus an additional $n^2 p \leq n^{2.5}$ cost for the orthogonal transformation of the variable space, plus the cost of solving the $p \times p$ system of quadratics. The latter is limited to $O(p)$ iterations which each cost $p^3/6$ multiplication and additions, so it is an insignificant $O(p^4) \leq 0(n^2)$ cost. The case $p=1$ is the most frequent in our computational experience, and in this case the quadratic equation is solved or minimized analytically. Thus solving the derivative tensor model costs essentially the same as finding the root of the standard linear model (1.3) by the QR factorization. It is possible to adapt the tensor solution algorithm to use the PLU factorization, or a sparse factorization, instead.

On singular problems with $\text{rank}(F'(x_*)) \geq n-p$, the solution of the tensor model by the above process usually will be well posed. The convergence analysis for singular systems of nonlinear equations shows that near x_*, we can expect the past steps $\{s_k\}$ to be in directions near the null vectors of $F'(x_*)$. In this case, the quadratic term of the tensor model supplies information in the directions where the linear model is lacking. This results in the linear system (2.7a) being well conditioned, and moves the ill-conditioning of the standard linear model into the linear term of the quadratic equations (2.7b), which still are well posed due to the quadratic term.

In addition, if $F'(x_c)$ happens to be singular or ill-conditioned at any iteration on any problem, and has p or fewer small or zero singular values, then the solution of the tensor model usually will be well posed for similar reasons.

2.3 Computational results with the derivative tensor method

A computer implementation of a derivative tensor method that is based upon the ideas summarized in Sections 2.1 and 2.2 has been extensively tested. A high level description of the method we have implemented is given in Algorithm 2.1.

Algorithm 2.1. An Iteration of the Derivative Tensor Method: given x_c, $F(x_c)$

1. Calculate $F'(x_c)$ and decide whether to stop. If not :
2. Select the past points to use in the tensor model from among the $\sqrt{n}$ most recent past points.
3. Calculate the second order term of the tensor model, T_c, so that the tensor model interpolates $F(x)$ at all the points selected in step 2.
4. Find the root of the tensor model , or its minimizer (in the l_2 norm) if it has no real root.
5. Select $x_+ = x_c - \lambda_c d_c$, where d_c either is the step calculated in step 4 or the Newton step, using a line search to choose λ_c.
6. Set $x_c \leftarrow x_+$, $F(x_c) \leftarrow F(x_+)$, go to step 1.

Details of our implementation are given in Frank [1984] and Schnabel and Frank [1984]. Note that the Newton step is calculated as a byproduct of the tensor model solution, and occasionally is used as the search direction in the tensor method. In particular, the Newton step is used in step 5 if Algorithm 2.1 finds a root d_T of the tensor model that isn't a descent direction

for $\|F(x)\|_2$ (a very rare occurrence in practice but not precluded in theory) and the point $x_c + d_T$ is unacceptable; if Algorithm 2.1 finds a minimizer of the tensor model at which the l_2 norm of the tensor model hasn't decreased enough from x_c; or if Algorithm 2.1 fails to find a root or minimizer of the tensor model in $8p$ iterations.

We compared our tensor method to an algorithm that is identical except that the second order term T_c always is zero. That is, the comparison algorithm is a finite difference Newton's method with a line search, except that the Newton step $-J_c^{-1}F(x_c)$ is modified to the approximation to the pseudo-inverse step $-(J_c^T J_c + \epsilon I)^{-1} J_c^T F(x_c)$ with ϵ small (see Dennis and Schnabel [1983]) when $J_c = F'(x_c)$ is singular or sufficiently ill-conditioned.

The Newton and tensor methods were compared on sets of nonsingular and singular test problems. The results are summarized briefly in Tables 2.1 - 2.3. The nonsingular test problems are a standard set in this field, given in Moré, Garbow, and Hillstrom [1981]; their dimensions range from $n = 2$ to 30. The singular problems are simple modifications of these problems constructed to have the same solution x_* with $\text{rank}(F'(x_*)) = n-1$ and $n-2$, respectively. The procedure for generating these singular problems is described in Schnabel and Frank [1984].

A significant feature of the test results is that the tensor method is virtually never less efficient than the standard method, and is almost always more efficient. In fact, on problems requiring ten or more iterations of the standard method, the tensor method always is more efficient. The gains in efficiency on the nonsingular problems are an average of about 18% if all test problems, including some very easy problems where no gains are likely, are considered, and an average of about 32% improvement on the harder problems. The gains in efficiency on the nonsingular problems are an average of about 40% and 30% in the rank $n-1$ and $n-2$ cases, respectively, and an average of about 57% and 46% improvement, respectively, on the

Table 2.1 -- Summary for Problems with $F'(x_*)$ Nonsingular

Problem Set	Number of Problems	Average Ratio, Tensor Method / Standard Method			Tensor Better	Standard Better	Tie
		Iterations	Jacobian evaluations	Function evaluations			
All problems	25	0.811	0.813	0.828	18	1	6
Harder problems only *	11	0.662	0.668	0.691	11	0	0

Additional problems solved by standard method only : 2
by tensor method only : 1

Table 2.2 -- Summary for Singular Test Set with Rank $(F'(x_*)) = n-1$

All Problems	17	0.576	0.609	0.603	15	0	2
Harder Problems Only *	9	0.392	0.429	0.434	9	0	0

Table 2.3 -- Summary for Singular Test Set with Rank $(F'(x_*)) = n-2$

All Problems	13	0.631	0.664	0.729	11	2	0
Harder Problems Only *	7	0.499	0.535	0.542	7	0	0

Additional problems solved by standard method only : 1
by tensor method only : 5

* Problems where slower method required at least 10 iterations

harder problems. In addition, the tensor method solved significantly more of the problems with rank$(F'(x_*)) = n-2$ than the standard method; this is not reflected in Table 2.3 which reflects only problems solved by both methods.

The improvements by the tensor method on the problems with rank($F'(x_*)$) = $n-1$ are partially explained by the faster local convergence of the tensor method as discussed in Section 2.4. In fact, our stopping tolerances were relatively loose; at tighter stopping tolerances the improvements by the tensor method on singular problems are greater. On nonsingular problems, the improvements by the tensor method apparently come from using a model that better interpolates $F(x)$; to our knowledge, the local convergence rate is no better than for Newton's method.

These computational results indicate that the derivative tensor method is consistently as reliable as currently used methods for solving systems of nonlinear equations. In addition, on problems where function evaluation is the dominant cost, it is consistently as efficient and often considerably more efficient, especially on problems with a small rank deficiency in $F'(x_*)$. The additional cost on the tensor method in arithmetic operations and computer storage is small. Indeed, in our tests, even on problems with $n = 30$, the number of past points interpolated by the tensor model generally was 1 or 2, so that the additional arithmetic and storage costs are very small. For these reasons, we believe that the derivative tensor method should be considered as a promising alternative to standard methods for general purpose software for solving systems of nonlinear equations.

2.4 Convergence analysis for the derivative tensor method

Frank [1984] has extensively analyzed the local convergence of the derivative tensor method. The most important result is that when rank($F'(x_*)$) = $n-1$, the method described in the previous sections is shown to be locally 3-step convergent with q-order 7/6, meaning that if $\|x_0 - x_*\|$ is sufficiently small, the sequence of iterates $\{x_k\}$ converges to x_* and, for some $c>0$, obeys

$$\|x_{k+3} - x_*\| \leq c \ \|x_k - x_*\|^{7/6}$$

for all $k > 0$. This rate of convergence is significantly faster than Newton's method which is linearly convergent with constant approaching 1/2 under the same assumptions. For simplicity of analysis, Frank's result is proven for a method that interpolates only the most recent past iterate (p=1); however it is not expected that interpolating additional past points would hurt its performance.

The reasoning behind the three step convergence result is interesting because it helps explain how the tensor method works on singular problems. From an arbitrary starting point close to x_*, it is shown that the first step provides at least linear convergence and results in an iterate whose error (its difference with x_*) is nearly in the direction of the null vector of $F'(x_*)$. The next step also provides at least linear convergence and also results in an error nearly in the null vector direction. Thus after two steps, the current iterate and the previous iterate are both close to being along the null vector direction from x_*, so that the step s_1 (the difference of these iterates) used in constructing the tensor term is essentially in this direction. Thus the quadratic term of the tensor model provides information in precisely the direction where the linear model is lacking. This causes the third step to be a fast one, in fact giving an order 1.5 improvement which would lead to three step q-order 1.5. (The smaller 7/6 rate comes from allowing for the possibility that the first or second steps are, by luck, too good.) After the third step, the error of the new iterate is not guaranteed to be close to the null space of $F'(x_*)$, so the three step process repeats. In practice, however, the errors appear to remain close to the null space so that one step, at least q-superlinear, convergence is observed.

When the rank of $F'(x_*)$ is less than $n-1$, the derivative tensor method probably is not faster than linearly convergent in theory because the model described in Section 3.1 does not approximate enough of $F''(x)$. However the test results of the previous section indicate that fast convergence still is obtained in the case rank$(F'(x_*)) = n-2$. It would be possible to

approximate the necessary portions of $F''(x)$ using previous values of the Jacobian rather than the function; this has not be pursued.

When $F'(x_*)$ is nonsingular Frank [1984] shows that the tensor method retains the q-quadratic convergence of Newton's method. This simply means that close to the solution, the quadratic term of the tensor model has a small effect and does not hurt the convergence. When $n=1$ it can be shown that the derivative tensor method has q-order 2.41, but the tensor method doesn't interpolate enough information for this result to extend to multi-dimensional problems.

3. Secant Tensor Methods

When analytic Jacobians are unavailable and function evaluation is sufficiently expensive, it is not cost effective to calculate a finite difference Jacobian approximation at each iteration of a method for solving systems of nonlinear equations. Instead, secant methods are used that only occasionally used finite difference Jacobians and otherwise base the method strictly upon the function values at the iterates.

The standard secant method for nonlinear equations, Broyden's method, uses a linear model of $F(x)$ around x_c that interpolates only $F(x_c)$ and $F(x_{-1})$, where x_{-1} is the previous iterate. Thus there are additional previous function values, namely $F(x_{-2})$, $F(x_{-3})$, ..., that a method still could interpolate. In the derivative tensor method discussed in Section 2, the interpolation of these function values was the basis for the tensor term T_c. In the secant case, however, since the first derivative matrix is not known, the value of $F(x)$ at a previous iterate x_{-k} only is sufficient to determine a linear model in the direction $x_c - x_{-k}$. Roughly speaking,

only if there are two previous iterates in the same direction from x_c is there sufficient information to determine a quadratic model. This indicates that it will be more difficult to form a quadratic model in the secant method case. Recall, however, that for singular problems the iterates often converge nearly along a single direction so that a quadratic model still may be possible.

Thus before considering how one might base a tensor secant model upon the interpolation of multiple function values, it is relevant to discuss how a linear secant model can interpolate multiple function values. We first summarize this briefly, and then discuss when and how we form a secant tensor model, and our computational results with the secant tensor method.

3.1 Linear Models with Multiple Secant Equations

Suppose x_c is the current iterate and x_{-1}, ..., x_{-p} are a set of p not necessarily consecutive past iterates chosen as in Section 2.1. That is, x_{-1} is the most recent past iterate, and the set of directions $\{s_k\} = \{x_{-k} - x_c\}$ are linearly independent. Then it is possible to choose the Jacobian approximation $J_c \in R^{n \times n}$ so that the secant model (1.2) interpolates $F(x_{-k})$, $k=1, \cdots, p$. This requires

$$F(x_{-k}) = F(x_c) + J_c s_k \quad , \quad k=1, \cdots, p \ . \tag{3.1}$$

If S, $Y \in R^{n \times p}$ are defined by column k of $S = s_k$, column k of $Y = F(x_{-k}) - F(x_c)$, then Schnabel [1983] shows that the closest matrix J_c to the previous Jacobian J_{-1} that causes (3.1) to be satisfied is

$$J_c = J_{-1} + (Y - J_{-1}S)\,(S^T S)^{-1}\, S^T \ . \tag{3.2}$$

Broyden's method simply is the special case of (3.2) with $p=1$. The update (3.1) appears to be a rank p change to J_{-1}, but if the linear model at the previous update interpolated all the

same previous function values except $F(x_c)$, then (3.2) is a rank one update.

Multiple secant updates along these lines have been proposed by Barnes [1965], Gay and Schnabel [1978], and Schnabel [1983]. Frank [1984] implemented a version where the past iterates to interpolate are chosen by the fairly restrictive criteria described in Section 2.1, essentially very strong linear independence plus only $\sqrt{n}$ iterates considered. Schnabel [1983] showed that this method is q-superlinearly convergent under the same conditions as Broyden's method. Frank found that on the nonsingular problems from the Moré, Garbow, and Hillstrom [1981] test set the multiple secant method was better on 10 problems, worse on 2, and about the same on 21, with an average improvement of 10%. On the singular problems described in Section 3.3, the multiple secant method was only marginally better than Broyden's method.

These results indicate that a properly implemented multiple secant method, using a linear model, is consistently at least as efficient as Broyden's method. Therefore, our secant tensor method, which also is based on interpolating multiple function values, builds upon this linear multiple secant model.

3.2 Forming the Secant Tensor Model

To determine a second order model of $F(x)$ using function values only, it is necessary to have more than $p+1$ function values that are nearly in some p dimensional subspace of the variable space. To illustrate the approach taken in forming our secant tensor model, suppose that there are two past iterates, x_{-1} and x_{-2}, and that the steps to them from x_c, s_1 and s_2, are nearly linearly dependent. That is, $s_2 = \alpha s_1 + z$ where $z^T s_1 = 0$ and $\|z\|/\|s_2\|$ is small. The tensor secant model (1.5) interpolates $F(x)$ at x_{-1} and x_{-2} if

$$F(x_{-k}) = F(x_c) + J_c s_k + \tfrac{1}{2} T_c s_k s_k \quad , \quad k = 1, 2 \ . \tag{3.3}$$

In the situation where s_1 and s_2 are nearly collinear, we interpret (3.3) as giving two pieces of information in the direction s_1 and no new information in the direction z. Thus we can make the a priori assumptions that

$$J_c z = J_{-1} z \quad , \quad T_c \; s_1 \; z = T_c \; z \; z = 0 \quad , \tag{3.4}$$

for we know that our minimum norm methods for choosing J_c and T_c will cause them to satisfy (3.4) if the secant equations are in the direction s_1 only. Combining (3.3) and (3.4) and using $s_2 = \alpha s_1 + z$ gives

$$F(x_{-1}) = F(x_c) + J_c s_1 + \tfrac{1}{2} T_c s_1 s_1 \tag{3.5a}$$

$$F(x_{-2}) = F(x_c) + \alpha J_c s_1 + J_{-1} z + (\alpha^2/2)\; T_c s_1 s_1 \quad . \tag{3.5b}$$

Equations (3.5) are two linear equations in the two unknown vectors $J_c s_1$ and $T_c s_1 s_1$, which are easily solved to yield

$$J_c s_1 = y \quad = (\alpha^2 u - v) \; / \; (\alpha^2 - \alpha) \tag{3.6a}$$

$$T_c s_1 s_1 = t \quad = (\alpha u - v) \; / \; (\alpha - \alpha^2) \tag{3.6b}$$

where $u = F(x_{-1}) - F(x_c)$, $v = F(x_{-2}) - F(x_c) - J_{-1} z$.

Equations (3.6) are the secant equations for J_c and T_c, respectively. Given these conditions, we form J_c as in the linear model multiple secant method and T_c as in the derivative tensor method. That is, J_c is given by (3.2) with $Y = y$ and $S = s_1$, while $T_c = a \; s_1 \; s_1$ with $a = t/(s_1^T \; s_1)^2$. Thus the tensor model becomes

$$M(x_c + d) = F(x_c) + J_c d + \tfrac{1}{2} \; a \; (w^T d)^2 \tag{3.7}$$

with $w = s_1$.

The remaining issue is the criterion for choosing s_2 to be "nearly linearly dependent" on s_1. The residual z cannot be allowed to be too large, or the inaccuracy in $J_c z$ may cause the

tensor T_c to be entirely inaccurate. Frank [1984] shows that it is necessary that $\|z\| \leq O(\|s_2\|^2)$ for the resultant T_c to be reliable. Furthermore, he indicates that one can expect consecutive iterates to satisfy this condition near the solution of a singular problem. There is no reason, however, to expect this condition to be satisfied for nonsingular problems. Thus it is likely that the quadratic term in the secant tensor method will be used near the roots of singular problems only.

Frank [1984] has generalized the above procedures to use more past iterates. First he chooses p strongly linearly independent directions to past iterates by the same process used in the derivative secant method and the linear model multiple secant method. This set of directions is the generalization of the direction s_1 in the above example. Then he chooses q directions to additional past iterates that are nearly dependent on the subspace spanned by the first set, in the sense described above. This set is the generalization of the direction s_2 in the above example. In our computational tests the second set contained 0 or 1 directions over 99% of the time, so it suffices to consider this case. The result is a set of equations similar to (3.5), which are easily reduced to the conditions $J_c S = Y$, $T_c ww = t$, for S, $Y \in R^{n \times p}$ and some w in the span of the columns of S. J_c then is chosen by (3.2) while $T_c = aww$ with $a = t/(w^T w)^2$. Thus the secant tensor model again is given by (3.7).

3.3 Solving the Secant Tensor Model

Algebraically, the secant tensor model (3.7) is just the special case of the derivative tensor model (2.5) with p=1. Thus its root or minimizer is found by the same procedures described in Section 2.2. Since the tensor term has rank one, the solution process results in finding the root or minimizer of one quadratic equation in one unknown followed by the solution of a system of $n-1$ linear equations and $n-1$ unknowns. Therefore no iterative procedure

is required and the cost is essentially the same as for finding the root of a linear model.

It is possible to perform each iteration of Broyden's method in $O(n^2)$ operations by sequencing a QR factorization of J_c as proposed originally by Gill and Murray [1972]. These efficiencies can be extended to the secant tensor method so that it does not cost appreciately more than the $O(n^2)$ implementation of Broyden's method. This was not done in our implementation.

In the case where the quadratic term of the secant tensor model has rank greater than one (which virtually never occurred in practice), Frank [1984] solves the secant tensor model by a minor generalization of the techniques of Section 2.2.

3.4 Computational Results with the Secant Tensor Method

A secant tensor method has been implemented and tested on the same nonsingular and singular problems that were used for the derivative tensor method tests described in Section 2.3. The basic iteration is summarized in Algorithm 3.1 below.

Algorithm 3.1. An Iteration of the Secant Tensor Method: given x_c, $F(x_c)$

1. Decide whether to stop. If not:
2. Select the two sets of past points to use in the tensor model from among the $\sqrt{n}$ most recent past points.
3. Calculate the first and second order terms of the tensor model, J_c and T_c, so that the tensor model interpolates $F(x)$ at all the points selected in step 2.
4. Find the root of the tensor model, or its minimizer (in the l_2 norm) if it has no real root.
5. Select x_+ by a trust region method that chooses x_+ to be a linear combination of the steepest descent direction, and the step calculated in step 4 or the root of the linear part

of the model.

6. Set $x_c \leftarrow x_+$, $F(x_c) \leftarrow F(x_+)$, go to step 1.

In addition to the strategy shown in Algorithm 3.1, the Jacobian is calculated by finite differences at the initial iteration and whenever the secant algorithm calculates two unsuccessful trial steps in a row. This later practice is taken from Moré's MINPACK algorithm (Moré, Garbow, and Hillstrom [1980]), as is the use of a trust region strategy at step 5.

The secant tensor method described above was compared to a Broyden's method version and to a linear model multiple secant method version of the same code. These were derived by setting $T_c = 0$ in the secant model, and by allowing one or multiple secant equations for the Jacobian approximation, respectively (i.e., $p=1$ or $p>1$ in (3.1) and (3.2)). The remainder of the code was unchanged.

On the nonsingular test problems from Moré, Garbow, and Hillstrom [1981], the secant tensor method was better than Broyden's method on 9 problems, worse on 5, and about the same on 18, with an average improvement in function evaluations of 9%. These results are marginally worse than the results for the linear model multiple secant method given in Section 3.1. Thus adding multiple interpolation conditions to the linear model seems to help a bit on nonsingular problems, but the tensor term seems to give no additional help.

On the test problems with rank $F'(x_*) = n-1$, the secant tensor method was better then Broyden's method on 18 problems, worse on 1, and tied on 4, with an average improvement of 25%. On the test problems with rank $F'(x_*) = n-2$, the secant tensor method was better than Broyden's method on 18 problems, worse on 1, and tied on 1, with an average improvement of 33%. In both of these cases, the linear model multiple secant method was not appreciably better than Broyden's method. Thus the addition of a tensor term seems to help considerably on problems with a low rank singularity at the solution.

In over 99% of the iterations on each test set, the rank of the tensor term was 0 or 1. Implementing a secant tensor method with a rank one tensor requires little additional storage and very few additional arithmetic operations in comparison to Broyden's method. Thus it appears that the gains mentioned above can be obtained at little additional cost to the computer, and a reasonably small increase in the complexity of the code. Therefore, such a secant tensor code might be a useful general purpose alternative to a Broyden's method code in a setting where singular or ill-conditioned problems are solved regularly.

4. Extensions of Tensor Methods to Optimization Problems

The nonlinear least squares problem

$$\min_{x \in R^n} \quad \|F(x)\|_2 \ , \ F : R^m \rightarrow R^n \tag{4.1}$$

can be viewed as an overdetermined version of the nonlinear equations problem (1.1). For nonlinear least squares, the Jacobian matrix always is computed analytically or approximated by finite differences. Thus it is natural to consider extending the derivative tensor method summarized in Section 2 to solving (4.1).

The derivative tensor method extends to overdetermined systems of equations with very little change. The formation of the tensor model is unchanged except that the model has m quadratic components rather than n. The solution procedure outlined in Section 2.2 now reduces the quadratic equations to the solution, in a least squares sense, of $m-n+p$ quadratic equations in p unknowns, followed by the solution of $n-p$ linear equations in $n-p$ unknowns. (If $p = 0$ this is just the QR algorithm for solving the linear model in the least squares sense.) The cost in arithmetic operations and computer storage again is hardly more than for a

standard method for nonlinear least squares.

A tensor method for nonlinear least squares along these lines currently is being implemented at the University of Colorado. This approach is most closely related to other nonlinear equations based approaches to nonlinear least squares. Of these the most computationally efficient appears to be Moré's trust region Levenberg Marquardt algorithm (Moré [1978]), implemented in MINPACK (Moré, Garbow, and Hillstrom [1980]), and it will be interesting to see how the tensor method compares to this on full rank and rank deficient problems. An alternate approach to nonlinear least squares, related more closely to viewing the problem as a special case of unconstrained optimization, is embodied in the NL2SOL algorithm of Dennis, Gay and Welsh [1981]. We also intend to compare the tensor method for nonlinear least squares to this approach.

The general unconstrained optimization problem is

$$\min_{x \in R^n} \quad f(x) : R^n \rightarrow R \quad . \tag{4.2}$$

The necessary condition for a solution of (4.2), $\nabla f(x) = 0$, is a system of n nonlinear equations in n unknowns. The standard local method for (4.2), also called Newton's method, simply is derived by applying (1.4) to this system of equations.

Thus it is tempting to expect that the tensor method for nonlinear equations can be applied to unconstrained optimization simply by applying the methods of Sections 2 and 3 to the system of equations $\nabla f(x) = 0$. This approach has several major flaws. One is that all the derivatives of the unconstrained optimization problem are symmetric, so that their approximation should be too, but the tensors T_c derived in Sections 2 and 3 are not 3-way symmetric. More importantly, if an unconstrained optimization problem has a singular Hessian matrix at a local minimizer, then the projection of the third derivative tensor $\nabla^3 f(x_*)$ in the null space direction v of the Hessian (i.e. $\nabla^3 f(x_*)\ v\ v\ v$) must also be 0. Thus approximating the third

derivative for unconstrained optimization, the analog of approximating the second derivative as is done in our tensor methods for nonlinear equations, would not be expected to help solve singular unconstrained optimization problems. It appears that approximations to both the third and fourth derivative matrices will be required to help solve singular optimization problems.

An approach to tensor methods for unconstrained optimization that makes small rank approximations to the third and fourth derivatives currently is underway at the University of Colorado. The effect of approximating both third and fourth derivative matrices is that, at each iteration, one must solve a small system of cubic equations in addition to the remaining linear equations. The storage and arithmetic overhead remains reasonable. Very preliminary computational results using this approach to solve singular optimization problems are encouraging.

5. References

J. Barnes [1965], "An algorithm for solving nonlinear equations based on the secant method", *Computer Journal* 8, pp. 66-72.

C. G. Broyden, J. E. Dennis Jr., and J. J. Moré [1973], "On the local and superlinear convergence of quasi-Newton methods", *Journal of the Institute of Mathematics and its Applications* 12, pp. 223-246.

D. W. Decker, H. B. Keller, and C. T. Kelley [1983], "Convergence rates for Newton's method at singular points", *SIAM Journal on Numerical Analysis* 20, pp. 296-314.

D. W. Decker and C. T. Kelley [1980a], "Newton's method at singular points I", *SIAM Journal on Numerical Analysis* 17, pp. 66-70.

D. W. Decker and C. T. Kelley [1980b], "Newton's method at singular points II", *SIAM Journal on Numerical Analysis* 17, pp. 465-471.

D. W. Decker and C. T. Kelley [1982], "Convergence acceleration for Newton's method at singular points", *SIAM Journal on Numerical Analysis* 19, pp. 219-229.

D. W. Decker and C. T. Kelley [1985], "Broyden's method for a class of problems having singular Jacobian", *SIAM Journal on Numerical Analysis* 22, pp. 566-574.

J. E. Dennis Jr., D. M. Gay, and R. E. Welsch [1981], "An adaptive nonlinear least-square algorithm", *ACM Transactions on Mathematical Software* 7, pp. 348-368.

J. E. Dennis Jr. and R. B. Schnabel [1979], "Least change secant updates for quasi-Newton methods", *SIAM Review* 21, pp. 443-459.

J. E. Dennis Jr. and R. B. Schnabel [1983], *Numerical Methods for Nonlinear Equations and Unconstrained Optimization,* Prentice-Hall, Englewood Cliffs, New Jersey.

P. D. Frank [1984], "Tensor methods for solving systems of nonlinear equations", Ph.D. Thesis, Department of Computer Science, University of Colorado at Boulder.

D. M. Gay and R. B. Schnabel [1978], "Solving systems of nonlinear equations by Broyden's method with projected updates", in *Nonlinear Programming 3,* O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, New York, pp. 245-281.

P. E. Gill and W. Murray [1972], "Quasi-Newton methods for unconstrained and optimization", *Journal of the Institute of Mathematics and its Applications* 9, pp. 91-108.

A. O. Griewank [1980a], "Analysis and modification of Newton's method at singularities", Ph.D. thesis, Australian National University.

A. O. Griewank [1980b], "Starlike domains of convergence for Newton's method at singularities", *Numerische Mathematik* 35, pp. 95-111.

A. O. Griewank [1985], "On solving nonlinear equations with simple singularities or nearly singular solutions", *SIAM Review* 27, pp. 537-563.

A. O. Griewank and M. R. Osborne [1981], "Newton's method for singular problems when the dimension of the null space is >1", *SIAM Journal on Numerical Analysis* 18, pp. 145-150.

A. O. Griewank and M. R. Osborne [1983], "Analysis of Newton's method at irregular singularities", *SIAM Journal on Numerical Analysis* 20, pp. 747-773.

H. B. Keller [1970], "Newton's method under mild differentiability conditions", *Journal of Computing and Systems Sciences* 4, pp. 15-28.

C. T. Kelley [1985], "A Shamanskii-like acceleration scheme for nonlinear equations at singular roots", preprint, Department of Mathematics, North Carolina State University.

C. T. Kelley and R. Suresh [1983], "A new acceleration method for Newton's method at singular points", *SIAM Journal on Numerical Analysis* 20, pp. 1001-1009.

J. J. Moré [1978], "The Levenberg-Marquardt algorithm: implementation and theory", in *Numerical Analysis, Dundee 1977, Lecture Notes in Mathematics 630,* G. A. Watson, ed., Springer-Verlag, Berlin, pp. 105-116.

J. J. Moré B. S. Garbow, and K. E. Hillstrom [1980], "User guide for MINPACK-1", Argonne National Laboratory Report ANL-80-74.

J. J. Moré B. S. Garbow, and K. E. Hillstrom [1981], "Testing unconstrained optimization software", *ACM Transactions on Mathematical Software* 7, pp. 17-41.

L. B. Rall [1966], "Convergence of the Newton process to multiple solutions", *Numerische Mathematik* 9, pp. 23-37.

G. W. Reddien [1978], "On Newton's method for singular problems", *SIAM Journal on Numerical Analysis* 15, pp. 993-996.

G. W. Reddien [1980], "Newton's method and high order singularities", *Comput. Math. Appl.* 5, 79-86.

R. B. Schnabel [1983], "Quasi-Newton methods using multiple secant equations," Technical Report CU-CS-247-83, Department of Computer Science, University of Colorado at Boulder.

R. B. Schnabel and P. Frank [1984], "Tensor methods for nonlinear equations", *SIAM Journal on Numerical Analysis* 21, pp. 815-843.